

Naval Medical Research Institute

8901 Wisconsin Avenue
Bethesda, MD 20889-5607

NMRI 95-62

October 1995



**MODIFICATION OF CANBERRA'S ACCUSPEC RADIATION
DATA COLLECTION PROGRAM TO PRODUCE
SPREADSHEET-FORMAT OUTPUT FILES**

D. R. Laws
P. Karnik
D. R. LaCaze
J. A. Novotny

Naval Medical Research
and Development Command
Bethesda, Maryland 20889-5606

Department of the Navy
Naval Medical Command
Washington, DC 20372-5210

Approved for public release;
distribution is unlimited

19951214 106

NOTICES

The opinions and assertions contained herein are the private ones of the writer and are not to be construed as official or reflecting the views of the naval service at large.

When U. S. Government drawings, specifications, or other data are used for any purpose other than a definitely related Government procurement operation, the Government thereby incurs no responsibility nor any obligation whatsoever, and the fact that the Government may have formulated, furnished or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise, as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

Please do not request copies of this report from the Naval Medical Research Institute. Additional copies may be purchased from:

National Technical Information Service
5285 Port Royal Road
Springfield, Virginia 22161

Federal Government agencies and their contractors registered with the Defense Technical Information Center should direct requests for copies of this report to:

Defense Technical Information Center
Cameron Station
Alexandria, Virginia 22304-6145

TECHNICAL REVIEW AND APPROVAL

NMRI 95-62

The experiments reported herein were conducted according to the principles set forth in the current edition of the "Guide for the Care and Use of Laboratory Animals," Institute of Laboratory Animal Resources, National Research Council.

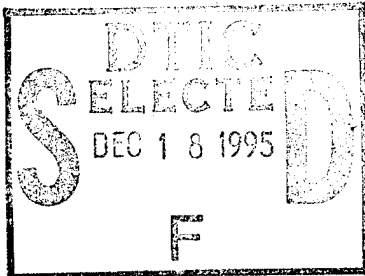
This technical report has been reviewed by the NMRI scientific and public affairs staff and is approved for publication. It is releasable to the National Technical Information Service where it will be available to the general public, including foreign nations.

THOMAS J. CONTRERAS
CAPT, MSC, USN
Commanding Officer
Naval Medical Research Institute

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE 1995	3. REPORT TYPE AND DATES COVERED Technical Report	
4. TITLE AND SUBTITLE Modification of Canberra's accuspec radiation data collection program to produce spreadsheet-format output files			5. FUNDING NUMBERS PE - 62233N PR - MM33P30 TA - 004 WJ - 1050	
6. AUTHOR(S) Laws DR; Karnik P; LaCaze DR; Novotny JA				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Medical Research Institute Commanding Officer 8901 Wisconsin Avenue Bethesda, Maryland 20889-5607			8. PERFORMING ORGANIZATION REPORT NUMBER NMRI 95-62	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Naval Medical Research and Development Command National Naval Medical Center Building 1, Tower 12 8901 Wisconsin Avenue Bethesda, Maryland 20889-5606			10. SPONSORING/MONITORING AGENCY REPORT NUMBER DN249500	
11. SUPPLEMENTARY NOTES				
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words)				
				
14. SUBJECT TERMS information systems, automated data collection, computer software modification			15. NUMBER OF PAGES	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT Unlimited	

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
<small>Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.</small>				
1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE October 1995	3. REPORT TYPE AND DATES COVERED Technical Oct 90 - Dec 91		
4. TITLE AND SUBTITLE Modification of Canberra's Accuspec radiation data collection program to produce spreadsheet-format output files.		5. FUNDING NUMBERS PE - 62233N PR - MM33P30 TA - .004 WU -1050		
6. AUTHOR(S) Laws, D.R., P. Karnik, D.R. LaCaze, and J.A. Novotny				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Medical Research Institute Commanding Officer 8901 Wisconsin Avenue Bethesda, Maryland 20889-5607		8. PERFORMING ORGANIZATION REPORT NUMBER NMRI 95-62		
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Naval Medical Research and Development Command National Naval Medical Center Building 1, Tower 12 8901 Wisconsin Avenue Bethesda, Maryland 20889-5606		10. SPONSORING / MONITORING AGENCY REPORT NUMBER DN249500		
11. SUPPLEMENTARY NOTES				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) This manuscript describes a modification of Canberra's Accuspec radiation data acquisition program to produce a compact, ASCII-format, concatenated output file with real-time display of data to a remote video terminal. This modification was driven by the needs of an experiment requiring the collection of data from several channels of input over many cycles, followed by statistical analysis and graphical display. These requirements were satisfied entirely within the framework of the AccuSpec Autosequence function: Only 2 of the 60 original source code files required modification and one C-language file was added to accomplish the necessary time-stamp recording, binary-to-ASCII conversion, data merging, concatenation, and display operations. The cycle reset time was 8 seconds. The modified program produces a single file, which is smaller by a factor of 1/375 th than the storage volume required for the original program's several hundred output files.				
14. SUBJECT TERMS information systems, automated data collection, computer software modification			15. NUMBER OF PAGES 22	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT Unlimited	

TABLE OF CONTENTS

	page
Acknowledgements	ii
Introduction	1
Methods	2
Hardware	2
Autosequence	2
Setup	3
Acquisition	3
Source code details	4
Modification #1: original file UTIL.C	4
Modification #2: original file MENU.C	5
Modification #3: created file NUPRINT.C	8
Discussion	8
Summary	9
References	9
Figure 1: AccuSpec Autosequence Code Summary with Modification in right margin	19

LIST OF APPENDICES

Appendix A: Autosequence	12
Appendix B: NUPRINT.C	15

Accession For	
NTIS CRAGI	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

DTIC QUALITY INSPECTED

ACKNOWLEDGEMENTS

This work was supported by NMRDC Work Unit No. 62233N MM33P30.004-1050.

The opinions expressed in this paper are those of the authors and do not reflect the official policy or position of the Department of Navy, Department of Defense, or the U.S. Government.

The authors thank Susan Mannix for expert editorial assistance.

INTRODUCTION

Scientific experiments and industrial procedures often require the cyclical acquisition of radiation data from any number of sources, and then storage or display of the data in a compact, easily managed format (1,2). One popular radiation data acquisition system is the Accuspec by Canberra. This system provides a means of performing cyclical data acquisition through its Autosequence feature, which is a high-level language for automated execution of Accuspec commands. One limitation, however, is that the system in its off-the-shelf form does not provide the means for merging or concatenating the collected data. Instead, for each cycle and every channel of input, the system produces one binary-format file, fixed at 5.1 kilobytes (kb) in size. For an experiment lasting, for example, 1 h with 5 channels collecting at a rate of 1 cycle/min, the Accuspec would produce 300 files requiring 1.5 megabytes (Mb) of storage space. Because many applications last several hours or even days, and may require more frequent acquisition, such high-volume output places a considerable burden on data storage capabilities. Also, organizing the data into a format suitable for display, mathematical manipulation, or analysis by other computer programs requires a number of file operations not available in the Accuspec system.

This manuscript describes a modification of the Accuspec program, which gives a compact, ASCII-format, concatenated output file containing the time-stamp and the data collected over many cycles from several channels of input. The requirements of the experimental work being performed in our laboratory will serve to illustrate a specific scenario for modification. However, the method employed by us was general enough to be readily adapted to a variety of requirements.

Our experiments, which form part of a radioactive gas kinetics physiology project (1), require the cyclical collection of radioactivity data from up to 5 scintillation detectors at a time. The radioactivity data is then analyzed with a nonlinear regression computer program, which requires as input a single ASCII file with one column for the time data and the remaining columns for the radioactivity data. We were able to achieve our goal of a single ASCII file entirely within the framework of the AccuSpec Autosequence by modifying only 2 of the 60 Accuspec source code files and adding 1 C-language file of our own to perform the necessary time-stamp recording, binary-to-ASCII conversion, data merging, concatenation, and other file operations.

METHODS

Hardware

Our hardware environment consisted of a Dell System 310 desktop computer (20 Mhz/386) with 4 Mb of RAM, a Canberra Accuspec A board containing the ADC, and a Western Digital, WD8003E Ethernet card. The external data collection equipment included a Canberra ND589 Analog Multiplexer (8 kb memory) capable of accepting up to 8 channels of input. The system was operated in the pulse-height analysis mode (PHA). Our remote video display was a DEC VT-100 terminal.

Autosequence

The Autosequence forming the framework of our data collection routine is shown in Appendix A and the functions performed by it are summarized in Fig. 1. Its operation is described in the following paragraphs.

Setup

The Autosequence begins by requesting user-supplied input (lines 9-16) for the desired counting time per cycle and the total number of cycles to execute. Then, the commands preparing each detector in the system for collection (lines 20-48) are executed to configure the system memory for the desired portions of the detected radioactivity spectrum and the cycle time b . The loop counter is set to 0, the loop limit is set to the desired number of cycles, and the system is prepared for video display (lines 50-57). With these setup tasks completed, the cyclical portion of the autosequence (lines 20-51) commences upon user command (line 53).

Acquisition

Once the required setup is in place, to begin the collection loop (lines 59-77), the system prepares the first detector to collect according to preset configuration files (line 59), clears the system memory (line 60), and then begins collecting data (line 61). This operation repeats in succession for the remaining detectors (lines 63-77) at a rate of approximately once every half-second.

The *wait\adc* command (line 79) suspends the autosequence during data acquisition until some preset condition is met, which in our case is the preset cycle time. The autosequence then executes the data transfer commands (lines 81-94), followed by a command for the required file operations (line 97). These segments of the Autosequence contain the 3 required modifications, described below.

Source code details

Modification #1: original file UTIL.C

The first of the required modifications involved the UTIL.C file. The added *util\print\ntotals* command opens a file for each detector and writes the data to disk. This is a modification of the original command *util\print\totals*. The files produced by the original command are in binary format and include a large volume of header information (date/time, group number, sample identification, configuration data, calibration information, measurement units, input device, in addition to the collected data), which is not required for our application. Although the Accuspec program provides a variety of utility commands, such as *lotcnv*, which performs binary-to-ASCII conversion, other problems persist. Most notably, the header information and numerous data files -- in the context of several hundred output files -- pose considerable file reduction and concatenation problems in addition to the mentioned storage problems. Performing all of these operations in real-time would add an unacceptable level of inefficiency and delay. Additional to these storage and format problems, the Accuspec system does not provide a path for real-time display of information to a remote video terminal, short of suspending the Autosequence for the execution of an external DOS command of some sort.

Our first task was to find a way to eliminate the unwanted header information from the initial data files. The source code file UTIL.C contains the functions utilized by the *util* command set. Only two of these functions required modification. In keeping with our philosophy of preserving the original workings of the program, we added two functions containing our code to this file instead of modifying the existing ones. Specifically, the

UTIL.C code was altered to produce a file with reduced header information along with the elapsed live-time and collected data for each detector. The choice of which header information to preserve is arbitrary, as none of it is used in our final output file; however, we wished to preserve a semblance of the format used in the Accuspec program.

Modification #2: original file MENU.C

The second modification involved the MENU.C file, which controls the available menu commands. We modified it to enable the recognition of the added *util* command *ntotals*. The Accuspec system executes the commands given by the Autosequence by recognizing the first letter or "hot key" of the command. For example, the hot key *T* of the original program corresponds to the *totals* command. So to call the new command *ntotals* we added the hot key *N* to MENU.C, leaving the original program unchanged. The modified MENU.C file produced a modified user-interactive menu displayed at the bottom of the screen

Data	Regions	Status	Totals	Keys	Screen	Ntotals
-------------	----------------	---------------	---------------	-------------	---------------	----------------

instead of the original menu

Data	Regions	Status	Totals	Keys	Screen
-------------	----------------	---------------	---------------	-------------	---------------

To execute the *ntotals* menu option, we added the function *uti_pri_totnew* to MENU.C. This produces the following description of the function in quotation marks at the bottom of the screen when *N* is selected. The C-language code for this:

```
uti_pri_totnew_des = {"Prints totals data as ASCII text file "}
```

The *uti_pri_totnew* function finishes executing the command by calling the functions *mf_print_ntotals* and *print_data_header* to produce the desired ASCII output file. We named these two functions using the same conventions as the originals *mf_print_totals* and *print_data_header*.

Modification #3: created file NUPRINT.C

File operations

With the collected data in ASCII format for each channel, we needed a means for merging the data and concatenating through successive cycles. To do this, we utilized an existing menu option, *uti\run*, which enables the execution of a DOS command during Autosequence operation. Our name for the new executable command, NUPRINT (code shown in Appendix B), is displayed in quotation marks with its required argument variables as letters on line 97 of the Autosequence.

NUPRINT opens each data file created by the *uti\print\ntotals* commands, writes the information to memory, closes the file (lines 39-49 for detector #1), and then performs any desired operations on the collected data. In our case, this involves calculating counts-per-minute (cpm) in each roi from the summed counts and the elapsed cycle live-time (line 50). This happens sequentially within NUPRINT for every channel of input. The DOS time is retrieved and converted to seconds-from-midnight, an arbitrary choice (line 119). The arguments passed from the Autosequence to NUPRINT enable the user to specify any

required variables for data processing before writing to file (lines 120-121). The final file operation is to open, append, and write the time and cpm data to the output file, which is closed before control is returned to the Accuspec system (lines 122-128).

Data display

One additional feature required for our application was to display the output data to a remote video terminal to enable real-time feedback during the experiments. This was accomplished by temporarily storing the output data of the current cycle in a write-only file and then making a DOS system call to copy this file to the COM1 port (line 145). The result is a cyclically updated display of the ongoing data collection process.

Return to Autosequence

Once the file and display operations are completed, control is returned to the Accuspec program and the Autosequence. The data acquisition then continues until a preset number of cycles is reached (Autosequence line 100), at which time the Autosequence jumps out of the loop and notifies the user that acquisition is complete. A portion of a sample output file is shown below.

Cycle#	time (seconds)	roi11, (cpm...	roi12,	...	roi51,	roi52 ...)
1,	65087,	501.,	720.,	...,	605.,	308.,
2,	65160,	511.,	740.,	...,	625.,	338.,
3,	65235,	521.,	750.,	...,	645.,	368.,
...						
...						
...						

Included are cycle number, time in seconds (from midnight), and partial data for 5 detectors and 2 rois per detector, respectively.

DISCUSSION

The modification of the Accuspec radiation data acquisition system described here is similar to an increasing number of similar modifications being made with other commercial software programs (3-6). The modification presented here enables data collection with a cycle reset time of only 8 s compared, for example, with the 22-second reset time of Canberra's earlier 35-PLUS system operating with a DEC PDP-1134 mainframe computer. The output file produced by the modified program for a 10-hour experiment producing 3,000 rows of data is about 40 kb versus the 15-megabyte output of the original program for the same task. So the modified program uses storage memory at a rate $1/375^{\text{th}}$ of the original. The modified program will accommodate a variety of conceivable applications. The NUPRINT program is just one example of several possibilities created to satisfy our particular requirements.

Although the final form of the modified program required only few and relatively simple modifications, the decisions by which we arrived at this particular version are worth considering in more detail. For the data manipulation and file operations, other means are available among the Accuspec commands for performing some of the required steps. For example, some of the data transfer functions performed by *move\data* could have been modified in the MOVEDATA.C file to include merging and concatenating operations. The TIME.C file, called by other Accuspec commands to provide DOS time, is not available directly as a user-accessed command, and therefore would have required modification. Thus, all of the alternatives that we could identify for accomplishing our goal would have entailed extensive changes to the original program. The MENU.C code would also have required

numerous changes in order to add several new commands, and would have generated a menu display altered considerably from the original. By contrast, our modifications produced just one additional menu item while leaving the remainder of the program unchanged.

We encountered some difficulties routing the required data to a remote video display. Our initial attempts involved transmitting the data via the Dell system communication port COM1. However, by mapping the memory addresses of the program, the Accuspec board, and the Dell COM1 port using Check-It (Touchstone), we found several memory address conflicts between the Accuspec and the host system, which cause the Dell COM1 port to be unavailable in Dell system as delivered. Specifically, in the presence of the Accuspec A board, IRQ#3 (Interrupt Request, a function of the PC-BIOS) is assigned to the Accuspec board communications port instead of to the Dell COM1 port. We used a "break-out" box to confirm that the Accuspec board port was available, and then connected a serial cable from it to the DEC VT-100. This enabled real-time viewing of the sequentially displayed time and data from each detector, in addition to the original spectrum display on the Accuspec host terminal. As a footnote, we found other conflicts between the Accuspec program and the Ethernet card, but these did not interfere with our application.

SUMMARY

The requirement for a file with easily sequenced time, summed counts per roi, and real-time viewing of data from several input channels was achieved through only a few modifications of the original AccuSpec program. The result was an ASCII file containing time and data in a spreadsheet format. The output file produced by the modified program for

a 10-hour experiment producing 3,000 rows of data was only 40 kb versus the 15-megabyte output of the original program for the same task. This format enables a variety of data processing options, such as statistical analysis, curve-fitting, graphing, data archiving, and transfer for use in other computer programs.

REFERENCES

1. Novotny, J.A., Mayers, D.L., Parsons, Y-F J., Survanshi, S.S., Weathersby, P.K., and Homer, L.D., "Xenon kinetics in muscle are not explained by a model of parallel perfusion-limited compartments." *Journal of Applied Physiology*, 68(3):876-890, 1990.
2. James, J.C., "A sample computer system for physiological data acquisition and analysis." *Computers in Biology and Medicine*, 20(6):407-413, 1990.
3. Britcher, J.B., Reengineering software- a case study. *IBM Systems Journal*, 10(1):10-20, 1990.
4. Ratzlaff, K.L., "Consideration of the need for custom data acquisition software." *Abstracts of Papers of the American Chemical Society*, 12:20-23, 1987.
5. Arnold, R.S., "Software restructuring." *Proceedings of the IEEE*, 31:41-45, 1989.
6. Anderson, K.J., Reuse of software modules. *American Telephone and Telegraph Technical Journal*, 13:45-47, 1988.

APPENDIX A Autosequence

```
1.      !NMRI.A      counting jobstream
2.      !
3.      ! This jobstream assumes the following are present in the system:
4.      !   1. 5 configurations named DET1, DET2, DET3, DET4, and DET5.
5.      !   2. 5 ROI files named NMRI1, NMRI2, NMRI3, NMRI4 and NMRI5.
6.      !
7.      ! Prompt the operator for specific inputs
8.      !
9.      ask u, " enter output file name ","out.dat"
10.     ask v, " enter value for b11","0"
11.     ask w, " enter value for b12","0"
12.     ask x, " enter value for krspl","0"
13.     ask y, " enter value for xespl","0"
14.     ask z, " enter value for ratio","0"
15.     !
16.     ask b,"Enter real time (1:00)","1:00"
17.     ask c,"How many cycles do you want to count ?","1"
18.     !
19.     acquire\context "DET1"
20.     acquire\erase
21.     acquire\preset\real b
22.     display\roi\purge
23.     move\roi\restore "NMRI1.ROI"
24.     !
25.     acquire\context "DET2"
26.     acquire\erase
27.     acquire\preset\real b
28.     display\roi\purge
29.     move\roi\restore "NMRI2.ROI"
30.     !
31.     acquire\context "DET3"
32.     acquire\erase
33.     acquire\preset\real b
34.     display\roi\purge
35.     move\roi\restore "NMRI3.ROI"
36.     !
37.     acquire\context "DET4"
38.     acquire\erase
39.     acquire\preset\real b
40.     display\roi\purge
41.     move\roi\restore "NMRI4.ROI"
42.     !
```

```

43.  !
44.    acquire\context "DET5"
45.    acquire\erase
46.    acquire\preset\real b
47.    display\roi\purge
48.    move\roi\restore "NMRI5.ROI"
49.  !
50.    let d="0"    !file name counter
51.    let e=c      !loop counter
52.  !
53.    ask f, "Press <ENTER> to begin.," "ENTER"
54.  !
55.  :loop
56.  !
57.    display\roi\advance
58.  !
59.    acquire\context "DET1"
60.    acquire\erase
61.    acquire\on/off\on
62.  !
63.    acquire\context "DET2"
64.    acquire\erase
65.    acquire\on/off\on
66.  !
67.    acquire\context "DET3"
68.    acquire\erase
69.    acquire\on/off\on
70.  !
71.    acquire\context "DET4"
72.    acquire\erase
73.    acquire\on/off\on
74.  !
75.    acquire\context "DET5"
76.    acquire\erase
77.    acquire\on/off\on
78.  !
79.    wait\adc    !wait until sample is done counting
80.  !
81.    acquire\context "DET1"
82.    util\print\totals "det1.tot"
83.  !
84.    acquire\context "DET2"
85.    util\print\totals "det2.tot"
86.  !

```

```

87.      acquire\context "DET3"
88.      util\print\totals "det3.tot"
89.      !
90.      acquire\context "DET4"
91.      util\print\totals "det4.tot"
92.      !
93.      acquire\context "DET5"
94.      util\print\totals "det5.tot"
95.      !
96.      inc d                      !increment file name counter
97.      util\run "nuprint ~v ~w ~y ~u ~d"
98.      !
99.      dec e                      !decrement # of cycles
100.     test\eq e,out             !finished if last cycle
101.     test\gt e,loop            !loop if more cycles to go
102.     :out
103.     ask f,"All cycles are counted, any key to continue","ENTER"
104.     !
105.     ! Counting is complete. Now ask if data should be erased from disk.
106.     !
107.     :alldun

```

APPENDIX B NUPRINT.C

```

1.  /* program to consolidate det1.tot, det2.tot,... det4.tot into det.tot */

2.  #include <stdio.h>
3.  #include <string.h>
4.  #include <dos.h>
5.  #include <math.h>
6.  #include <io.h>
7.  #include <fcntl.h>
8.  #include <bios.h>
9.  #include <sys\types.h>

10. FILE *stream1,*stream2,*stream0, *stream3, *stream4, *stream5;
11. char line [80],outf[20];
12. char str1[10],str2[10],str3[10],colon;
13. int result;
14. struct dostime_t ptime;
15. long hour1,min1,hour2, min2;
16. float etime11,etime12,etime21,etime22,etime31,etime32;
17. float etime41,etime42,etime51,etime52;
18. float r11,r12,r21,r22,r31,r32,r41,r42,r51,r52;
19. long count11,count12,count21, count22, count31, count32, count41, count42;
20. long count51, count52;
21. unsigned comstat;
22. float b11,b12,krspl,xespl,ratio;
23. int loop;

24. main(argc,argv)
25. int argc;
26. char *argv[];

27. {   if (argc == 6)
28.     {
29.       sscanf (argv[1],"%f",&b11);
30.       sscanf (argv[2],"%f",&b12);
31.       sscanf (argv[3],"%f",&krspl);
32.       sscanf (argv[4],"%f",&xespl);
33.       sscanf (argv[5],"%s",outf);
34.       sscanf (argv[6],"%d",&loop);

35.     }
36.     else
37.       printf ("Incorrect number of arguments %d provided \nMUST BE 5\n",

```

```

38.     argc - 1);

39.     if(( stream1 = fopen ("det1.tot","r")) !=NULL )
40.     {
41.         result = fscanf(stream1,"%s %s %s %d %c %d",
42.             str1,str2,str3,&hour1,&colon,&min1);
43.         result = fscanf(stream1,"%s ",line);
44.         result = fscanf(stream1,"%s ",line);
45.         result = fscanf(stream1,"%s ",line);
46.         result = fscanf(stream1,"%ld %f",&count11,&etime11);
47.         result = fscanf(stream1,"%s ",line);
48.         result = fscanf(stream1,"%ld %f",&count12,&etime12);
49.         fclose (stream1);
50.         r11 = etime11/60.0;
51.         r12 = etime12/60.0;
52.     }
53.     else
54.         printf ("Problem opening det1.tot \n");

55.     if(( stream2 = fopen ("det2.tot","r")) !=NULL )
56.     {
57.         result = fscanf(stream2,"%s %s %s %d %c %d",
58.             str1,str2,str3,&hour2,&colon,&min2);
59.         result = fscanf(stream2,"%s ",line);
60.         result = fscanf(stream2,"%s ",line);
61.         result = fscanf(stream2,"%s ",line);
62.         result = fscanf(stream2,"%ld %f",&count21,&etime21);
63.         result = fscanf(stream2,"%s ",line);
64.         result = fscanf(stream2,"%ld %f",&count22,&etime22);
65.         fclose (stream2);
66.         r21 = etime21 / 60.0;
67.         r22 = etime22 / 60.0;
68.     }
69.     else
70.         printf ("Problem opening det2.tot \n");

71.     if(( stream3 = fopen ("det3.tot","r")) !=NULL )
72.     {
73.         result = fscanf(stream3,"%s %s %s %d %c %d",
74.             str1,str2,str3,&hour2,&colon,&min2);
75.         result = fscanf(stream3,"%s ",line);
76.         result = fscanf(stream3,"%s ",line);
77.         result = fscanf(stream3,"%s ",line);
78.         result = fscanf(stream3,"%ld %f",&count31,&etime31);

```

```

79.     result = fscanf(stream3,"%s ",line);
80.     result = fscanf(stream3,"%ld %f",&count32,&etime32);
81.     fclose (stream3);
82.     r31 = etime31 / 60.0;
83.     r32 = etime32 / 60.0;
84.     }
85.     else
86.     printf ("Problem opening det3.tot \n");

87.     if(( stream4 = fopen ("det4.tot","r")) !=NULL )
88.     {
89.     result = fscanf(stream4,"%s %s %s %d %c %d",
90.         str1,str2,str3,&hour2,&colon,&min2);
91.     result = fscanf(stream4,"%s ",line);
92.     result = fscanf(stream4,"%s ",line);
93.     result = fscanf(stream4,"%s ",line);
94.     result = fscanf(stream4,"%ld %f",&count41,&etime41);
95.     result = fscanf(stream4,"%s ",line);
96.     result = fscanf(stream4,"%ld %f",&count42,&etime42);
97.     fclose (stream4);
98.     r41 = etime41 / 60.0;
99.     r42 = etime42 / 60.0;
100.    }
101.    else
102.    printf ("Problem opening det4.tot \n");

103.    if(( stream5 = fopen ("det5.tot","r")) !=NULL )
104.    {
105.    result = fscanf(stream5,"%s %s %s %d %c %d",
106.        str1,str2,str3,&hour2,&colon,&min2);
107.    result = fscanf(stream5,"%s ",line);
108.    result = fscanf(stream5,"%s ",line);
109.    result = fscanf(stream5,"%s ",line);
110.    result = fscanf(stream5,"%ld %f",&count51,&etime51);
111.    result = fscanf(stream5,"%s ",line);
112.    result = fscanf(stream5,"%ld %f",&count52,&etime52);
113.    fclose (stream5);
114.    r51 = etime51 / 60.0;
115.    r52 = etime52 / 60.0;
116.    }
117.    else
118.    printf ("Problem opening det5.tot \n");

```

```

119.    _dos_gettime(&ptime);

120.    krspl = (.472679 - (0.001307*((count12-b12)*.0001)));
121.    ratio=1/(1-(krspl*xespl));

122.    if(( stream0 = fopen (outf,"a+"))  !=NULL )
123.    {
124.        fprintf (stream0,"%d, %u, %.0f., %.0f., %.0f., %.0f., %.0f., %.0f., %.0f.,
%.0f., %.0f.,\n",
125.            loop,((ptime.hour*3600)+(ptime.minute*60)+ptime.second),
126.            count11/r11,count12/r12,count21/r21,count22/r22, count31/r31,
127.            count32/r32,count41/r41, count42/r42, count51/r51, count52/r52);

128.    fclose (stream0);
129.    }

130.    else
131.    printf ("Problem opening output file \n");
132.    if(( stream0 = fopen ("this.jnk","w"))  !=NULL )
133.    {
134.        fprintf (stream0,"%d, %.0f. %.0f., %.0f. %.0f., %.0f. %.0f., %.0f. %.0f.,
%.0f. %.0f.,\n",
135.            loop,
136.            (((count11/r11)-b11)-(krspl*((count12/r12)-b12)))*ratio),
137.            (((count12/r12)-b12)-(xespl*((count11/r11)-b11)))*ratio),
138.            count11/r11, count12/r12,
139.            count21/r21, count22/r22,
140.            count31/r31, count32/r32,
141.            count41/r41, count42/r42,
142.            count51/r51, count52/r52);
143.    fclose (stream0);
144.    }
145.    system ("copy this.jnk com1:");
146.    }

```


Figure 1

AccuSpec Autosequence Code Summary with modifications in right margin

Autosequence Initialization

Name output file

Enter background, spillover, and ratio numbers

Enter count time and number of cycles to count

Get configuration for each detector

Erase old configuration/setup new configuration

Purge old ROI's

Install new ROI's

Initialize loop counter

Wait for user to initiate

Get ROI configuration for each detector

Erase old data from memory

Begin collecting data

Wait for collection cycle to end

Get data for each detector

{ UTIL.C modified to give ROI true counts

Print totals to disk using new menu function Ntotals

{ MENU.C modified to recognize

added command (Ntotals)

Increment loop counter

1) Open ntotals files for each detector

Run NUPRINT

2) Open output file for appending new data to old

3) Concatenate, print cycle number

4) Get DOS time and convert to seconds-from-midnight

5) Print current data to a temporary file and display on VT-100

6) Close all files

Decrement loop count number

Test for end of loop number

NO

YES

Inform user of completion

Terminate on user command